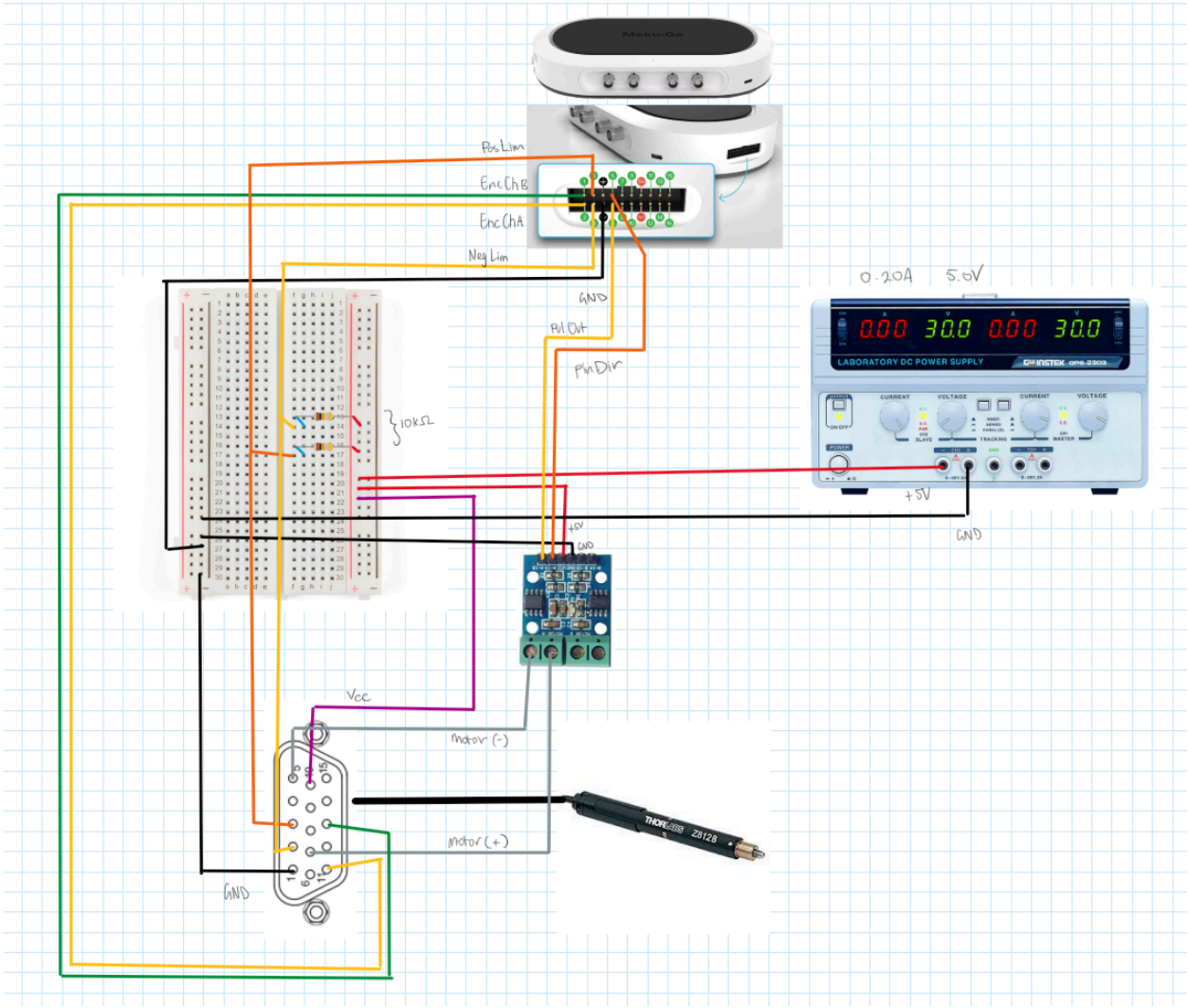
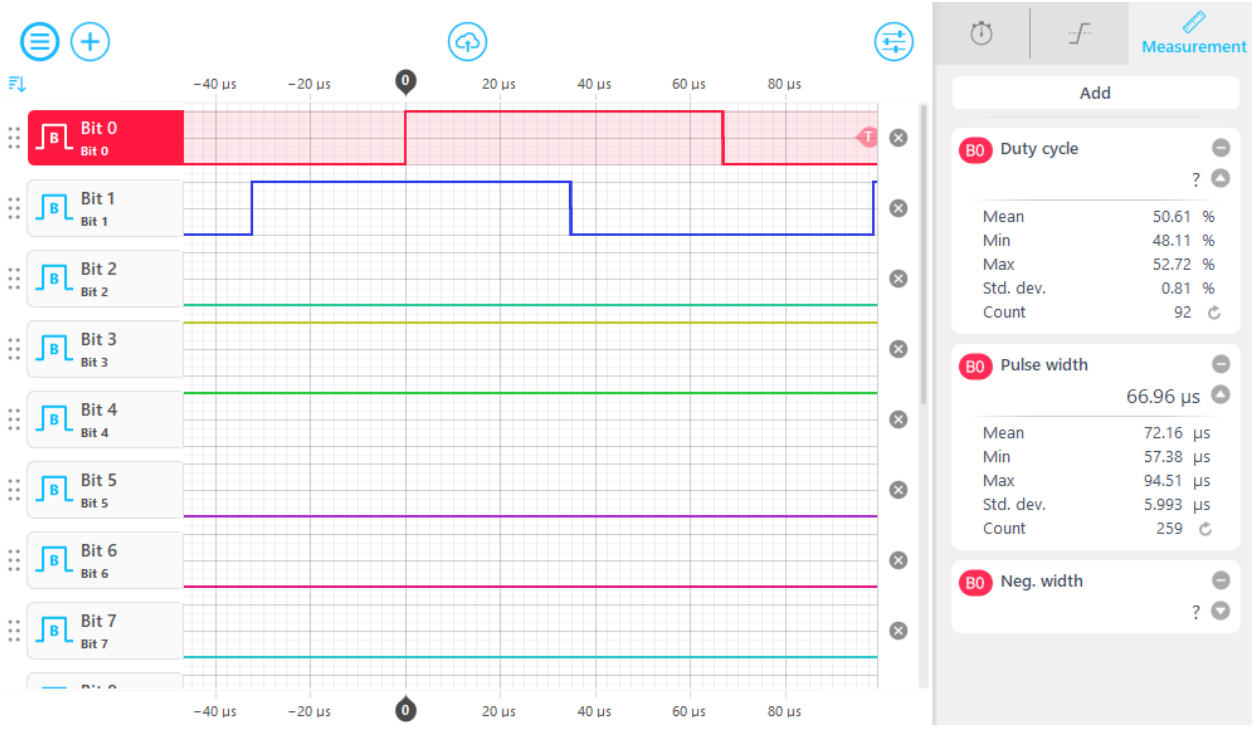
Actuator Testing



Rough Circuit Design of test setup.

1. Determine the time duration of the high drive signal for triggering one tick count from the encoder lines. For example, the actuator returns a tick when it receives 1 ms high signal. This could help us to determine the period and duty-cycle of the pulse train in the MCC.

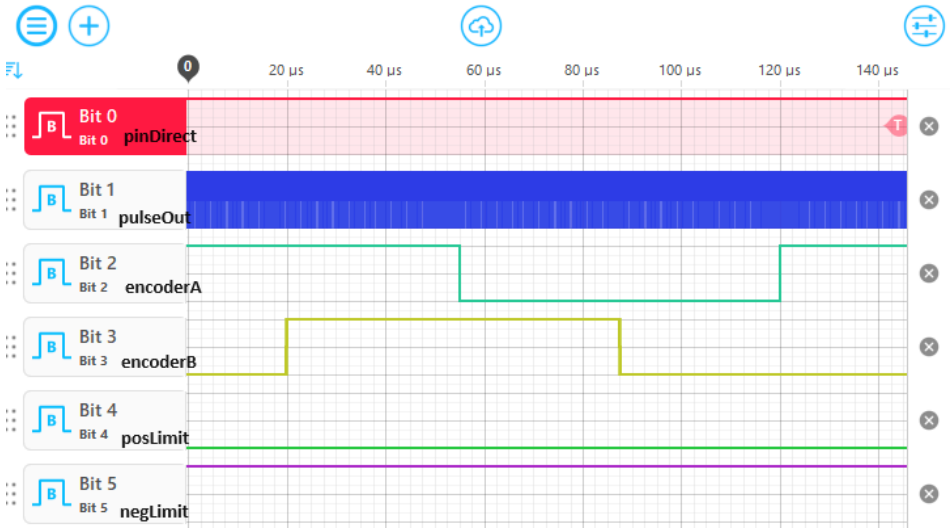


At 50% duty cycle, the encoder signals have a pulse width of 66.96 micro seconds.
It takes 69.96 microseconds to return a single tick.

2. Complete the MCC verification with Logic Analyzer to make sure that the MCC is working as expected.

Works as expected. Was able to add a way to view the number of counts in binary, viewed by the logic analyser. For every simulated pulse of the encoders, the counter incremented by 4.

3. Once the MCC has been verified, we can start testing the MCC on the actuator. We can set a long period and small duty-cycle in MCC, then set a fairly small target number of pulses. Then, we can connect the digital I/O pins to the actuator to verify the functionality of MCC.

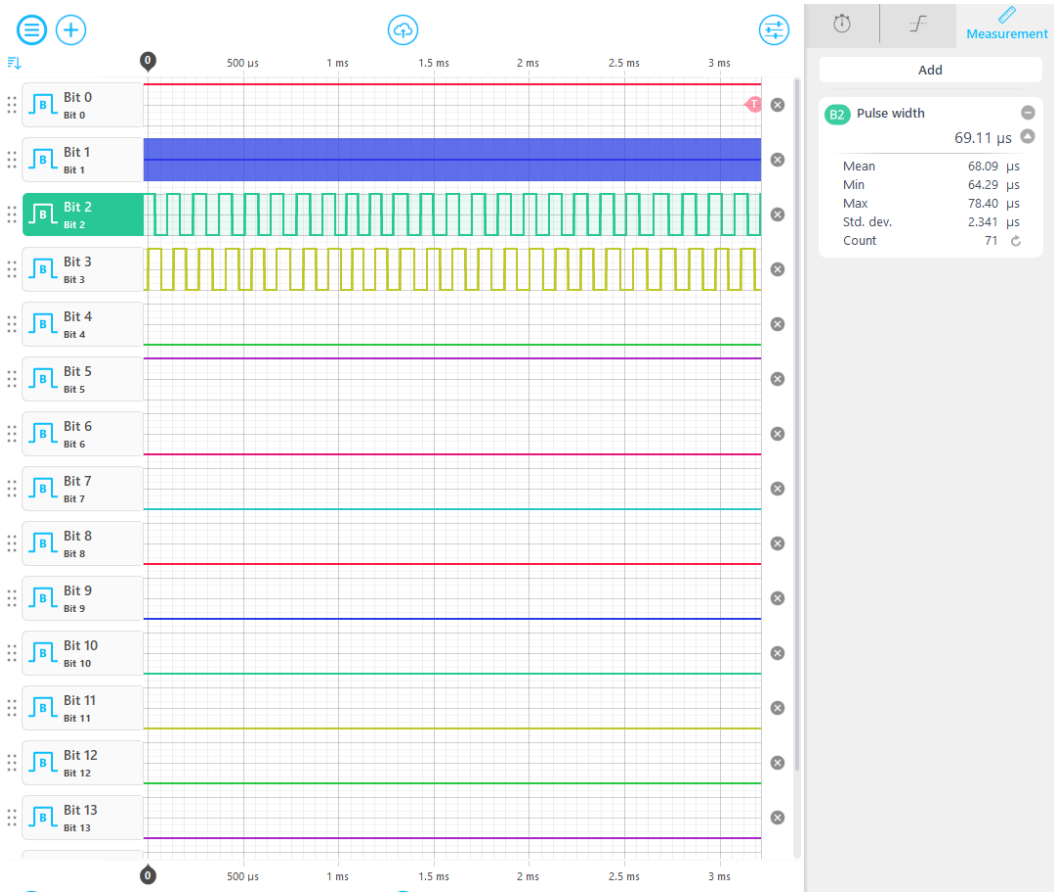


To move approximately 2.5mm, 28 encoder signals were counted. This was obtained using the following register values:

Register	Decimal (Unsigned)	Decimal (Signed)	Hexadecimal	Binary
Control0	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control1	589,824	589,824	0009 0000	0000 0000 0000 1001 0000 0000 0000 0000
Control2	524,296	524,296	0008 0008	0000 0000 0000 1000 0000 0000 0000 1000
Control3	524,296	524,296	0008 0008	0000 0000 0000 1000 0000 0000 0000 1000
Control4	67,108,864	67,108,864	0400 0000	0000 0100 0000 0000 0000 0000 0000 0000
Control5	327,688	327,688	0005 0008	0000 0000 0000 0101 0000 0000 0000 1000
Control6	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control7	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control8	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
				0000 0000 0000 0000

Auto commit

Control2 is set up to detect 8 edges, which is 4 pulses. When run for 2.5mm, there were a total of 28 pulses, which is 56 edges. I manually disengaged the actuator as it did not stop automatically after 4 pulses.



Even after 71 pulses, the actuator keeps moving until manually switched off.

4. The next step would be testing the negLimit and posLimit with an external pin or Pattern Generator to make sure that MCC does stop when one of the Limits is high.

5. Then, we can test the H-bridge to make sure that it could switch the direction of the actuator driving signal.

The H-Bridge can control direction. The H-Bridge I used during testing has 2 inputs, one can be used for speed by changing the duty cycle, and the other can be used to control the direction.

For 50% duty cycle, rotation speed in either direction is the same. However, if we use an 80% duty cycle for example, the motor will rotate at 80% of power in one direction, and 20% of power in the other direction.

6. After that, we can try a larger duty-cycle to make sure that the MCC could operate at a higher speed.

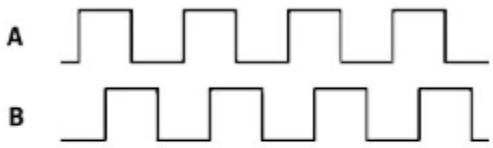
Debugging Session (11/04/2024)

Signal Testing

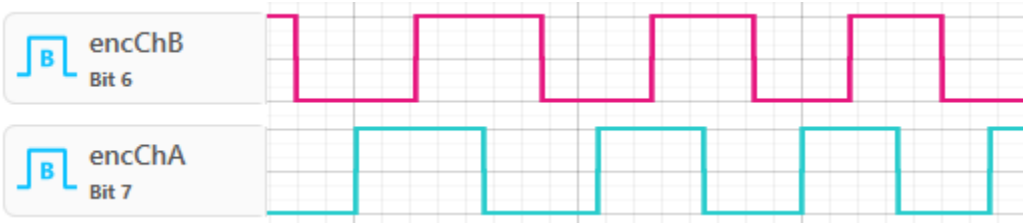
Limit Switches
Initially, testing the MCC with the limit switches connected was causing errors - this was likely because the Positive limit switch is physically broken, causing its signal to always be LOW while the Negative limit switch is always HIGH. The MCC has been written to observe a LOW on both limit switches as normal operation, and if either signal goes HIGH, it has reached one of the limits. However, in the data sheet, it says that the circuit is to be configured in a way which makes the switches HIGH unless triggered. This will be overcome by inverting the signals in MCC. Currently, tests have been conducted with the limit switches disconnected.

I will test the negative limit switch as this switch is still functional.

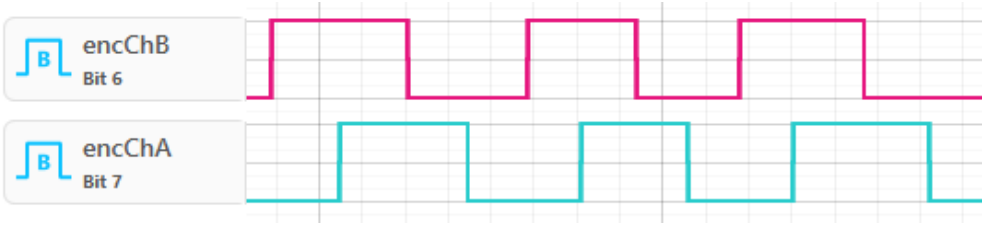
Encoder Channels



When **A** leads **B**, the actuator is driving forwards. When **B** leads **A**, the actuator is driving backwards. The counter will thus need to be able to resolve this offset.



Forward Operation: Channel A is leading Channel B

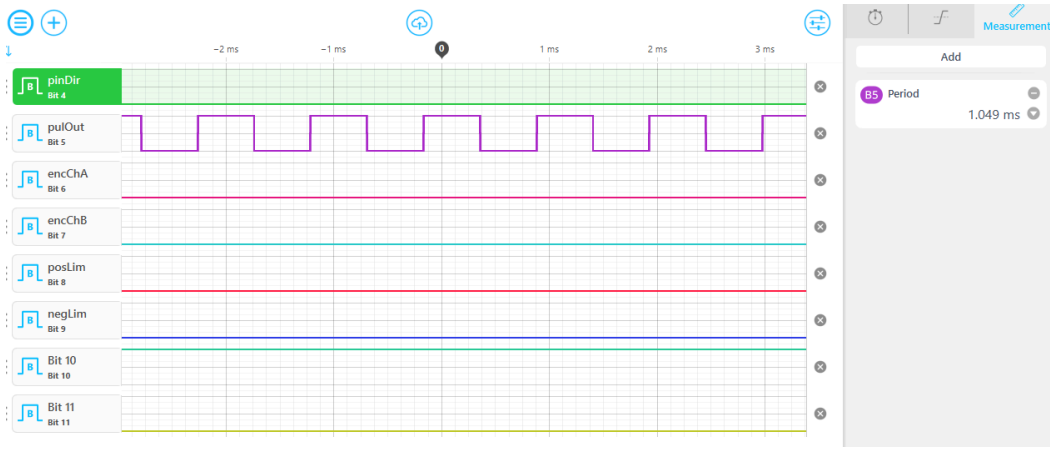


Reverse Operation: Channel B is leading Channel A

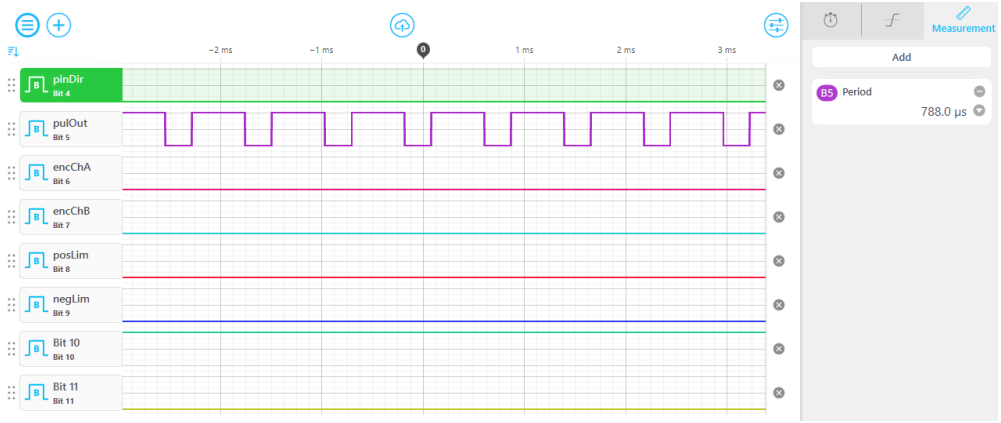
Pulse Out

Controlling of the duty cycle and period behaves as expected.

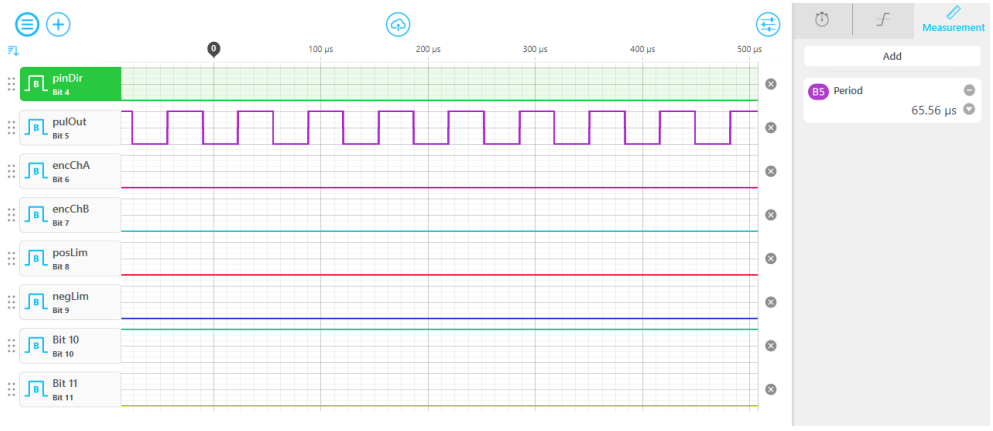
4000 4000 (50% duty cycle)



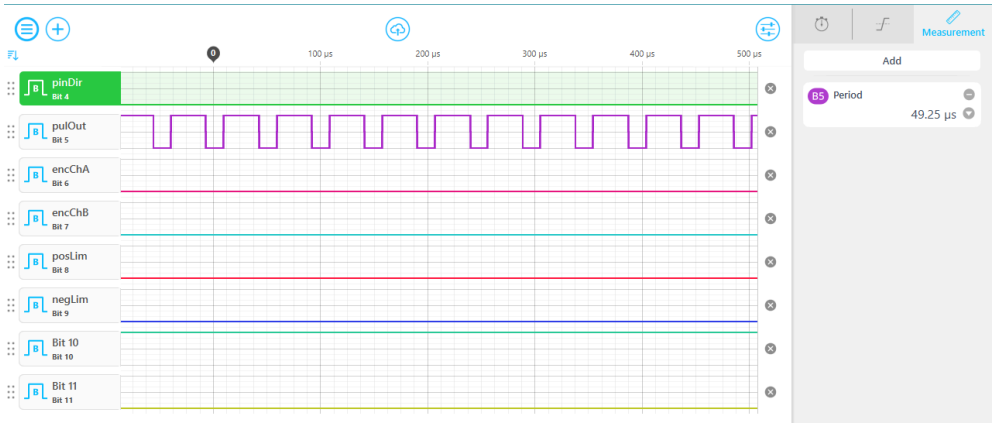
4000 2000 (66% duty cycle)



0400 0400 (50% duty cycle, but much smaller period)



0400 0200 (66% duty cycle, but much smaller period)



Direction

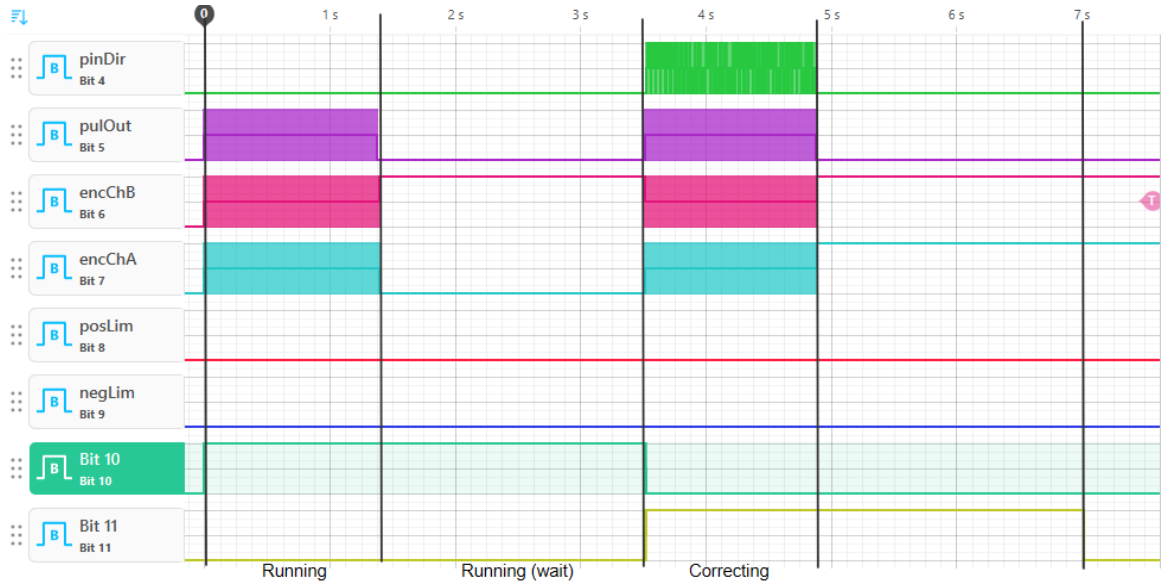
If the direction is HIGH, then the actuator will travel in the positive direction and if LOW, in the negative direction. This behaves as expected.

Importantly, if the duty cycle is 90% in one direction, the motor will drive to 90% power, but if the direction is reversed, it will only rotate at 10% power. The MCC needs to take this into account by inverting the Pulse out signal when the direction is changed, as to preserve the duty cycle and thus actuator speed. Otherwise, the duty cycle must be manually flipped every time the direction of operation is changed.

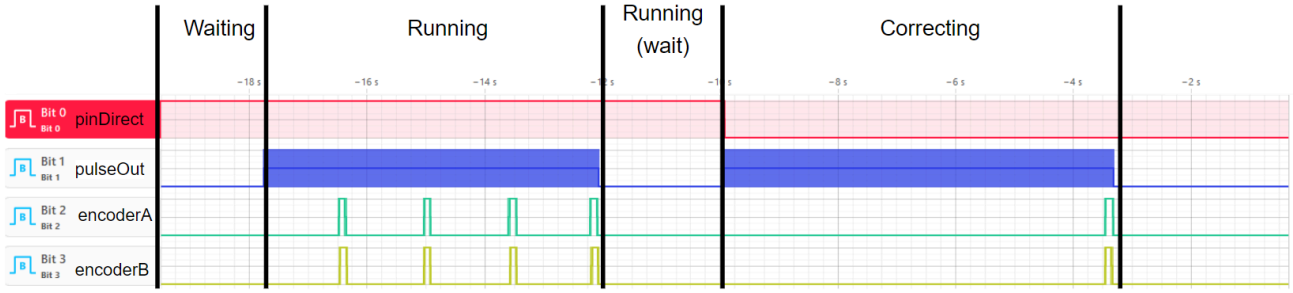
Operation

Register Values

Register	Decimal (Unsigned)	Decimal (Signed)	Hexadecimal	Binary
Control0	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control1	1,073,758,208	1,073,758,208	4000 4000	0100 0000 0000 0000 0100 0000 0000 0000
Control2	8,388,736	8,388,736	0080 0080	0000 0000 1000 0000 0000 0000 1000 0000
Control3	16,813,824	16,813,824	0100 8F00	0000 0001 0000 0000 1000 1111 0000 0000
Control4	67,108,864	67,108,864	0400 0000	0000 0100 0000 0000 0000 0000 0000 0000
Control5	2,097,160	2,097,160	0020 0008	0000 0000 0010 0000 0000 0000 0000 1000



We can see that the actuator undergoes operation, where the actuator is moving the desired number of pulses. After this point is reached, there is a brief period of waiting, and then correction begins. After a final waiting period, the actuator is ready for operation again and has entered the ‘waiting’ state indicated by bits 10 and 11 both being LOW. This follows the expected behavior, as simulated by Hank inside of the logic Analyser.



(Resolved - Baud Rate too high!)
I have found that there is strange behavior of the H-bridge, namely, when a duty cycle pulse is used to drive the motor, and direction is changed, the motor only drives in one direction. However when a binary input is used, operation in both directions can occur.
Could this be causing issues?

Case 1:
pulseOut = 1
pinDirect = 0
Outcome = Forward driven, Full speed

Case 2:
pulseOut = 50% duty cycle
pinDirect = 0
Outcome = Forward driven, half speed

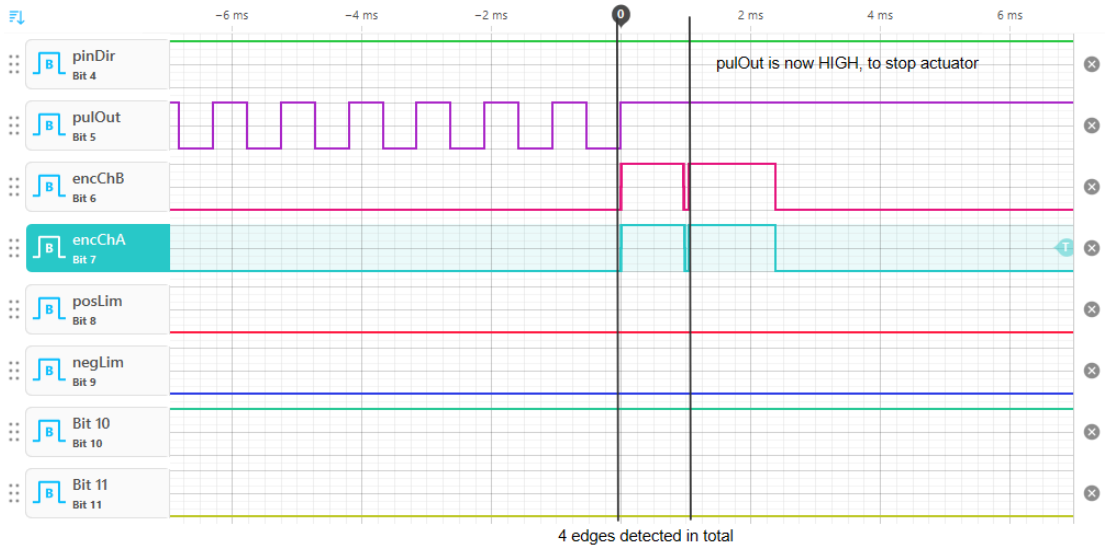
Case 3:
pulseOut = 50% duty cycle
pinDirect = 1
Outcome = Not driven

Case 4:
pulseOut = 0
pinDirect = 1
Outcome = Reverse driven, Full speed

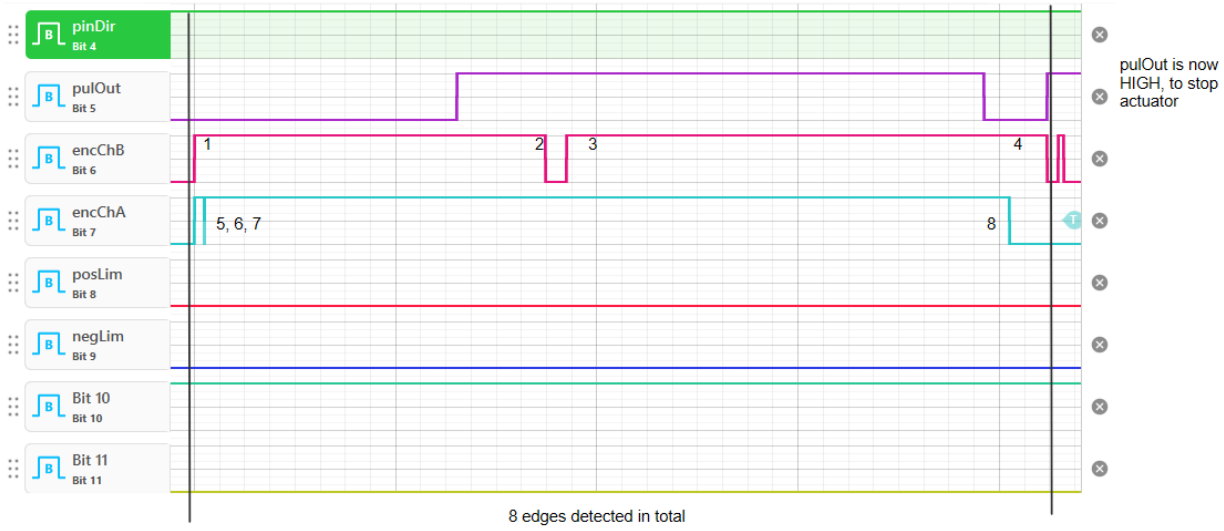
Travel

Initially, we found that regardless of the set number of pulses to look for, the actuator tended to travel the same distance which was close to 1 mm.

Hank made some refinements to the code, so it can now more accurately count the number of encoder edges and halt the output Pulse - stopping the actuator. When considered for a small number of pulses, the actuator cannot be seen moving, however using the logic analyser we can observe the number of pulses and at what point the actuator's driving signal is cut.For instance, for a desired number of 4 edges, the following output was observed.



For a desired number of 8 edges,



Further Testing (12/04/2024)

Hank's suggestions

1. I realized that the pulse number was not -8 when we set the pulseNum as 8008, it was -(2^16 - 8) instead. The pulseNum is a signed 16-bit value, so we need to calculate the 2's complement code for the negative numbers.
Changing the pulseNum to 2's complement does correctly drive the actuator in the intended direction.

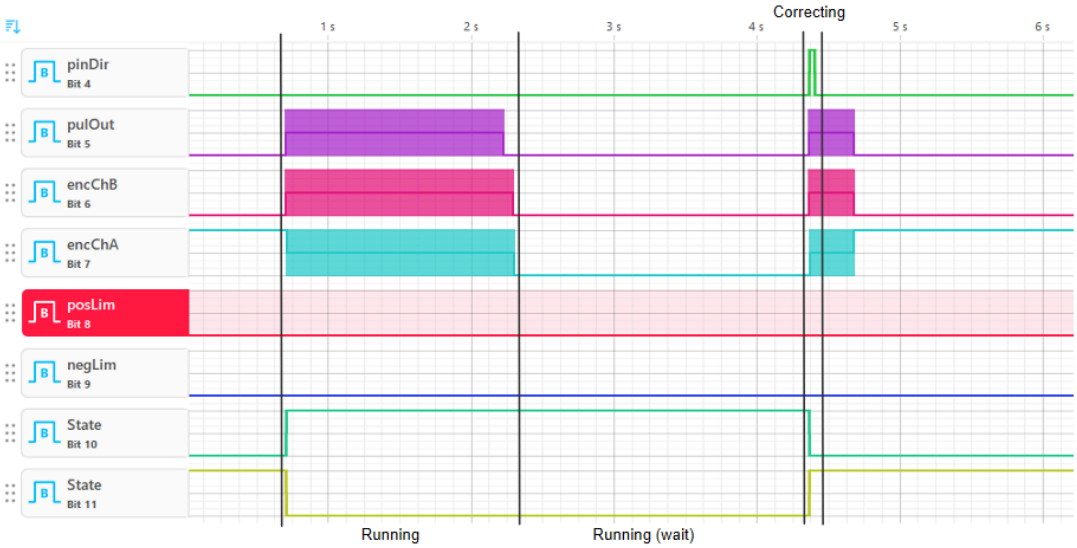
For larger values of pulseNum I observed that the actuator can be seen moving now, and travels larger distances. Importantly, the direction of rotation can be observed, and matches the expectations - positive number moves the actuator forward, negative number moves the actuator backward.

2. The correction stage could have a lot of fluctuations due to the long activeIntervalCorrect. The correction stage is expected to move the actuator with a low speed. Would you please **try some small activeIntervalCorrect values**? For example, 100 or other values below 400. And would you please **also try some large deadIntervalCorrect values**? For example, 1000 or even 4000. One thing to note is that this low duty cycle might not even drive the actuator at all, so please feel free to change the values. **The key is that we need to decrease the driving speed in the correction stage to limit the fluctuations.**

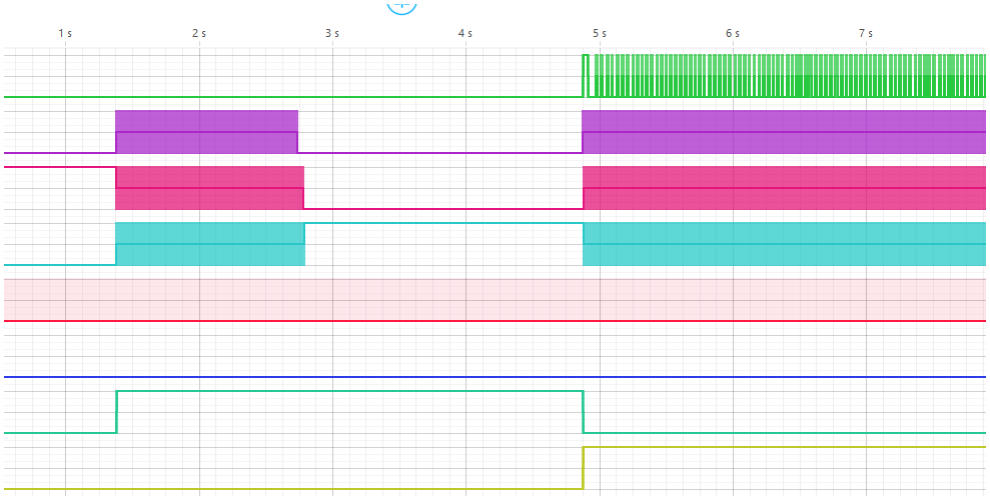
These register values were used for the following test.

Register	Decimal (Unsigned)	Decimal (Signed)	Hexadecimal	Binary
Control0	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control1	1,073,758,208	1,073,758,208	4000 4000	0100 0000 0000 0000 0100 0000 0000 0000
Control2	33,558,528	33,558,528	0200 1000	0000 0010 0000 0000 0001 0000 0000 0000
Control3	36,864	36,864	0000 9000	0000 0000 0000 0000 1001 0000 0000 0000
Control4	67,108,864	67,108,864	0400 0000	0000 0100 0000 0000 0000 0000 0000 0000
Control5	327,688	327,688	0005 0008	0000 0000 0000 0101 0000 0000 0000 1000

With activeIntervalCorrect = 200 and deadIntervalCorrect = 1000 (16% Duty cycle)



With activeIntervalCorrect = 400 and deadIntervalCorrect = 1000 (29% Duty cycle)



If the Duty cycle is > 23% (approx), the correcting interval is much longer as the actuator is moving too quickly to correct accurately.

If the Duty cycle is < 23%, like in the first screen shot (16%), then the correction stage is very quick because the actuator is moving slowly and is more accurate.

For a general rule of thumb, keeping duty cycle less than 20% will ensure the correction phase is more accurate.

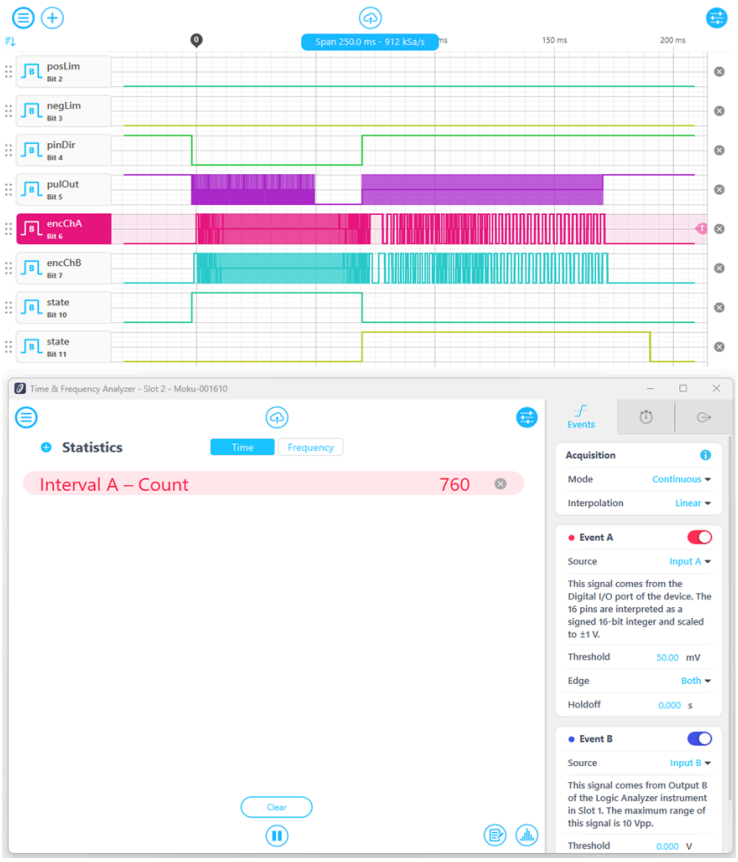
Accuracy Testing

Using the Time and Frequency Analyser (TFA), the exact number of Encoder signals was able to be counted. A series of tests were conducted, varying the value of pulseNum over a range of values between 10 and 100,000. The results are documented in the table below.

Test no.	pulNum	Count encChA	Approx Combination	Difference	%
1	-10	27	54	-44	440
2	10	27	54	-44	440
3	-100	148	296	-196	196
4	100	90	180	-80	80
5	-500	453	906	-406	81.2
6	500	302	604	-104	20.8
7	-1000	760	1520	-520	52
8	1000	542	1084	-84	8.4
9	-5000	2744	5488	-488	9.76
10	5000	2554	5108	-108	2.16
11	-10000	5228	10456	-456	4.56
12	10000	5222	10444	-444	4.44
13	-20000	10351	20702	-702	3.51
14	20000	10424	20848	-848	4.24
15	-50000	25448	50896	-896	1.792
16	50000	25310	50620	-620	1.24
17	-100000	50774	101548	-1548	1.548
18	100000	50101	100202	-202	0.202

Note that the other register values were kept fixed, as follows.

Register	Decimal (Unsigned)	Decimal (Signed)	Hexadecimal	Binary
Control0	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control1	536,887,296	536,887,296	2000 4000	0010 0000 0000 0000 0100 0000 0000 0000
Control2	33,558,528	33,558,528	0200 1000	0000 0010 0000 0000 0001 0000 0000 0000
Control3	5,000	5,000	0000 1388	0000 0000 0000 0000 0001 0011 1000 1000
Control4	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control5	8	8	0000 0008	0000 0000 0000 0000 0000 0000 0000 1000
Control6	5	5	0000 0005	0000 0000 0000 0000 0000 0000 0000 0101
Control7	625,000	625,000	0009 8968	0000 0000 0000 1001 1000 1001 0110 1000
Control8	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control9	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control10	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control11	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control12	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control13	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control14	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
Control15	0	0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000



This image shows how the waveforms and count were recorded.

The TFA is measuring both rising and falling edges for the encChA, so count will be the total number of edges detected for a single channel. If we double this, that will give the approximate number of total edges detected. By subtracting count from pulseNum, we will obtain the amount of additional edges (which could be from over-travel and/or correcting). Since we do not currently have a way to count the net number of edges detected, the closest way to approximate the number of edges traveled is by minimizing the correction pulses. This can be achieved by reducing the duty cycle to minimize overshoot. This however is only practical for reasonably small values of numPulse, as it can take up to 15 seconds to move 10,000 pulses with a low duty cycle.

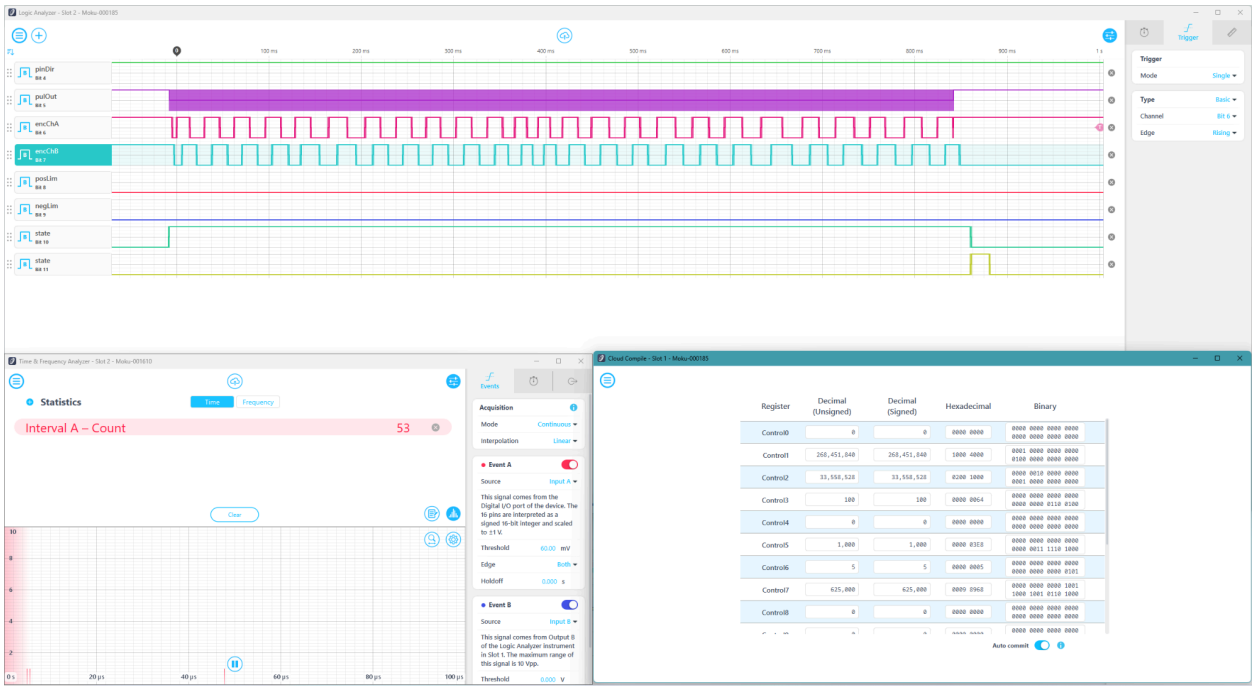
We can see from the data that the actuator tends to travel a distance which is closer to the expected distance when it is traveling forward. This is purely down to the behavior of the experimental setup. The actuator travels slower in one direction than the other when using a duty cycle to control speed (even with a 50% duty cycle). The reduced speed in the forward direction results in the MCC being able to more accurately control the actuator.

There is also a very large error associated with a small number of pulses, thus a very low duty cycle would be recommended. The value of numPulse should be no less than 1000, and a duty cycle of 20% would be typical for steady operation. Note that the tests were carried out with a

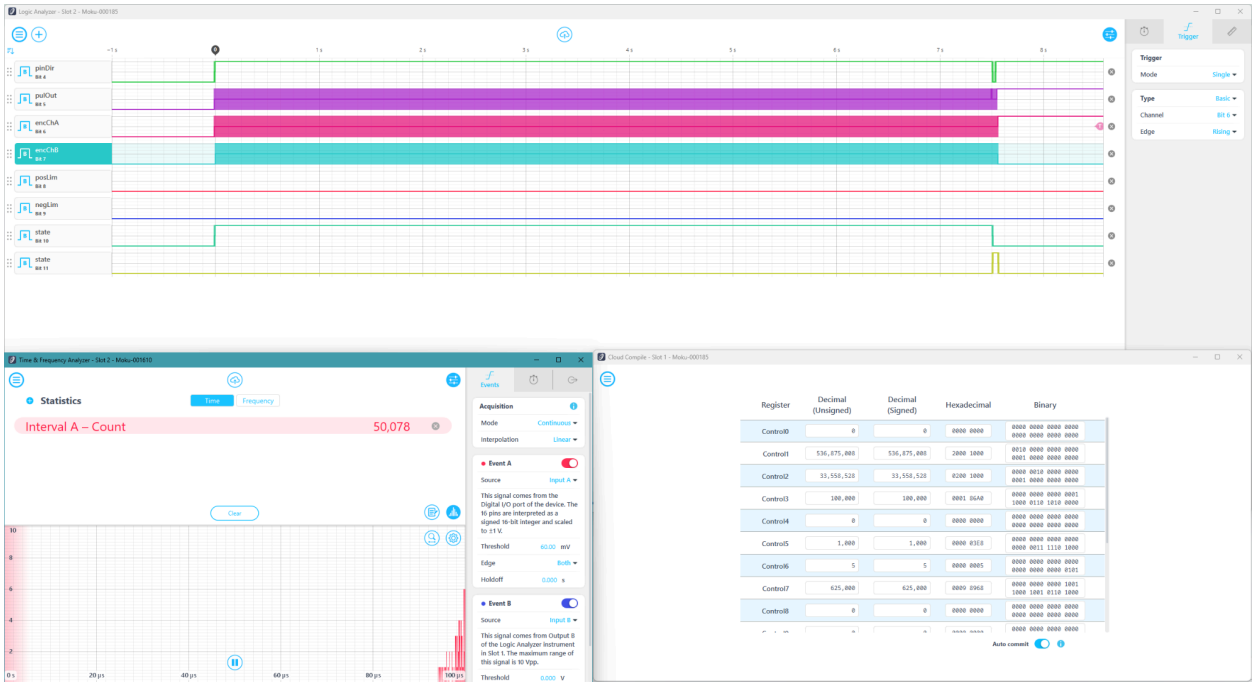
33% duty cycle. We observe that for large values of numPulse, the overshoot/correction only contributes to a small percentage of the edges detected.

From an operational point of view, I can confirm that the actuator behaves as expected. Tripling the numPulse value makes the actuator travel 3 times further, halving duty cycle makes the actuator move half as quick, and changing the direction of pulseNum changes the direction of travel.

Test case 1.



Test case 2.


Logic Analyzer - Slot 2 - Moku-000185
pinDir
pulOut
encChA
encChB
posLim
negLim
state
state
Trigger
Mode
Single
Type
Basic
Channel
Bit 6
Edge
Rising
Time & Frequency Analyzer - Slot 2 - Moku-001610
Statistics
Time
Frequency
Interval A – Count
50,078
Clear
Events
Acquisition
Mode
Continuous
Interpolation
Linear
Event A
Source
Input A
This signal comes from the Digital I/O port of the device. The 16 pins are interpreted as a signed 16-bit integer and scaled to ±1 V.
Threshold
60.00
mV
Edge
Both
Holdoff
0.000
s
Event B
Source
Input B
This signal comes from Output B of the Logic Analyzer instrument in Slot 1. The maximum range of this signal is 10 Vpp.
Threshold
0.000
V
Cloud Compile - Slot 1 - Moku-000185
Register
Decimal (Unsigned)
Decimal (Signed)
Hexadecimal
Binary
Control0
Control1
Control2
Control3
Control4
Control5
Control6
Control7
Control8
Auto commit